Lending Risk Analysis for a Lending Company

Project Brief:

A consumer finance company named 'XYZ' specialises in lending various types of loans to urban customers. When the company receives a loan application, it must decide for loan approval based on the applicant's profile.

There are two types of risks are associated with the bank's decision:

1. If the applicant is likely to repay the loan, company doesn't approve the loan results in a loss to the company.
2. If the applicant is not likely to repay the loan (likely to be defaulter), company approves his loan may also lead to a financial loss to the company.

Based on the dataset supplied company wants to understand the driving factors (or driver variables) behind loan default, i.e. the variables which are strong indicators of default.

Step-1: Data Cleaning & Manipulation:

Python Libraries Used:

```
##Importing relevant packages
import pandas as pd
import numpy as np
from datetime import datetime
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime
```

- Created a function to eliminate columns/rows having more than 50% missing values.
- Created a function to remove single value unique columns from the data frame.
- Removed duplicate rows (if any)
- Made a list of unrequired columns from the dataframe and drop that whole list later on.

- Dropped the rows with ‘Nan’ values if their percentage was found below 10%.
- Some columns found with more than 90% values as ‘0’ zero, hence tackled their ‘Nan’ values by replacing them by ‘0’.
- However, some columns with string values and containing ‘Nan’ also were replaced by ‘undisclosed’ keyword (since we don’t want to miss out the information associated with these rows).
- Changed to data type of some columns to desired data types (int→ float).
- Made a function to Strip-Off the white spaces and convert from lowercase to Uppercase (and vice-versa) depending on the requirements.

1st Challenge I faced:

The date of Loan_issue as per the dataset must vary from 1946 to 2008 but while checking it with the column, it was found out to be 1969 to 2068.

So, I had to figure out a way to modulate the current range into the desirable range.

Solution:

Subtracted 100 from the year column (Loan_issue) where the value is greater than 2008 based on the above output.

```
df_loan.loc[df_loan['earliest_cr_line_year'] > 2008 , 'earliest_cr_line_year']  = df_loan[df_loan['earliest_cr_line_year'] > 2008]['earliest_cr_line_year']-100
```

```
array([1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977, 1978, 1979,
       1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988, 1989, 1990,
       1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001,
       2002, 2003, 2004, 2005, 2006, 2007, 2008, 2046, 2050, 2054, 2056,
       2059, 2061, 2062, 2063, 2064, 2065, 2066, 2067, 2068], dtype=int64)
```



```
array([1946, 1950, 1954, 1956, 1959, 1961, 1962, 1963, 1964, 1965, 1966,
       1967, 1968, 1969, 1970, 1971, 1972, 1973, 1974, 1975, 1976, 1977,
       1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987, 1988,
       1989, 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999,
       2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008], dtype=int64)
```

Step-2: Data Analysis (Univariant + Bivariant):

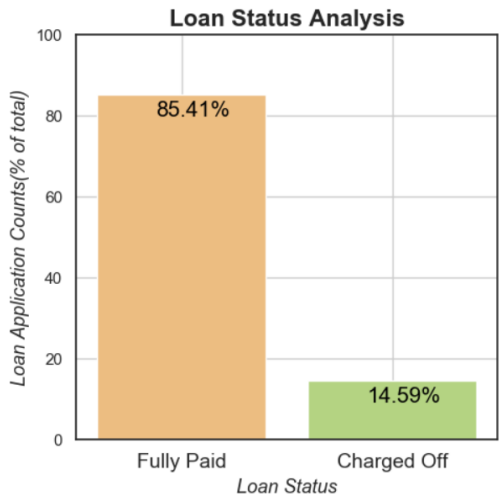
Here on we made the 'df_loan' dataframe as we have already created a copy in the 'master_f_loan' dataframe.

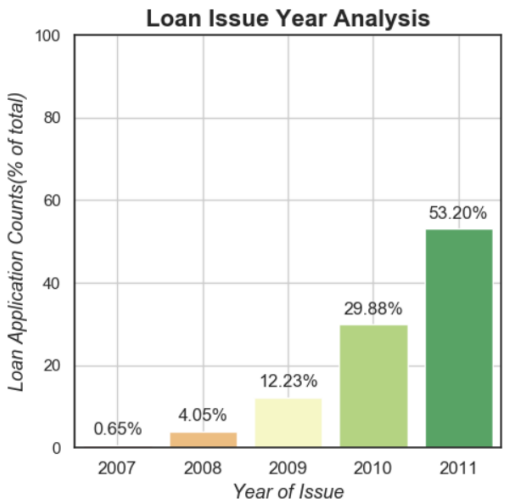
A glimpse of Segmented Univariate Analysis:

Loan Status Analysis:

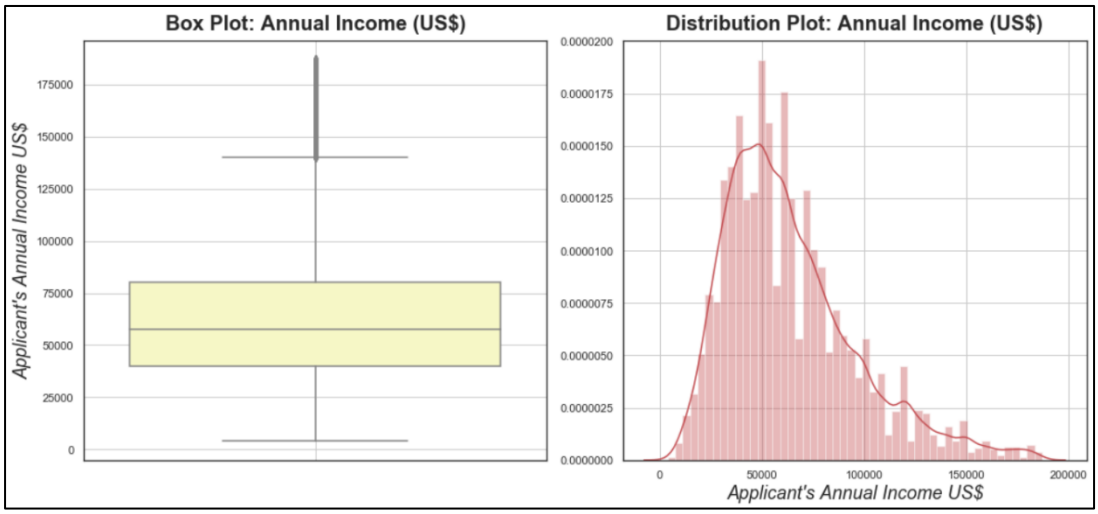
```
##Lets check how 'loan_status'(%) is distributed
sns.set(style='white')
plt.figure(figsize=(5,5), dpi=100)
br = pd.DataFrame(df_loan.loan_status.value_counts(normalize=True).mul(100).sort_values())
br.reset_index(inplace=True)
br = br.rename(columns= {'loan_status':'loan_status_count'})

ax = sns.barplot(x='index',y='loan_status_count', data=br, palette='RdYlGn')
ax.set_xticklabels(labels=['Fully Paid','Charged Off'], rotation='horizontal', fontsize=14)

plt.xlabel('Loan Status', fontsize= 13, fontstyle='italic')
plt.ylabel('Loan Application Counts(% of total)', fontsize= 13, fontstyle='italic')
plt.ylim(0,100)
plt.title('Loan Status Analysis', fontsize=16, fontweight='bold')

for i in ax.patches:
    # get_x pulls left or right; get_height pushes up or down
    ax.text(i.get_x()+0.28, i.get_height()-5.5, \
            str(round((i.get_height()), 2))+"%", fontsize=15, color='black',\
            rotation=0)

plt.grid(True)
plt.tight_layout()
plt.show()
```

- Overall Default Rate stands at 14.59%.
- Approval rate of loans jumped by around 78% from the year 2010 to 2011.
- Approval Rate of loans is higher during the Holiday Season (November & December).

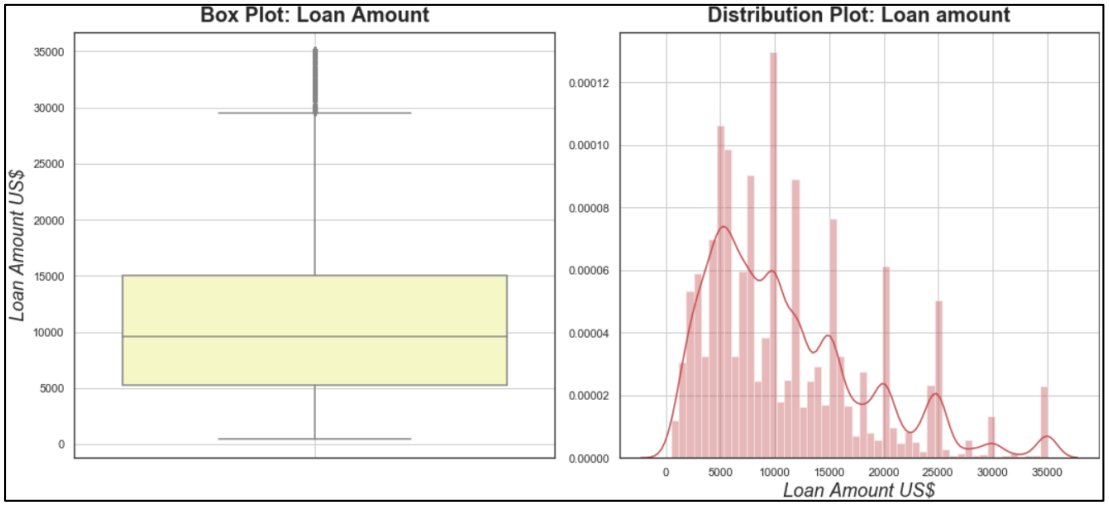




```
plt.figure(figsize=(15,7))
sns.set(style='white')

plt.subplot(1, 2, 1)
sns.boxplot(y = df_loan[df_loan.annual_inc <= 187000 ]['annual_inc'],palette='RdYlGn')
plt.title("Box Plot: Annual Income (US$)", fontsize= 20, fontweight='bold', pad=10)
plt.ylabel("Applicant's Annual Income US$", fontsize= 18, fontstyle='italic')
plt.grid(True)


plt.subplot(1, 2, 2)
sns.distplot(df_loan[df_loan.annual_inc <= 187000 ]['annual_inc'] , hist= True ,color="r")
plt.title("Distribution Plot: Annual Income (US$)", fontsize= 20, fontweight='bold', pad=10)
plt.xlabel("Applicant's Annual Income US$", fontsize= 18, fontstyle='italic')
plt.grid(True)

plt.tight_layout()
plt.show()
```



- 50% of the applicants earn between 40K USD and 82K USD annually.
- Annual Income was highly skewed as expected; ignoring the outliers, the average salary of applicants is 68.78K USD.
- 50% of loan applicants request a loan amount between 5.3K USD and 15K USD.



2nd Challenge I faced:

Just to figure out dependencies (if any), wanted to plot and analyse the dependencies with the help of "Pearson Correlation Coefficients". I wanted plot this via correlation heatmap indicating the correlation among several features of the data frame.

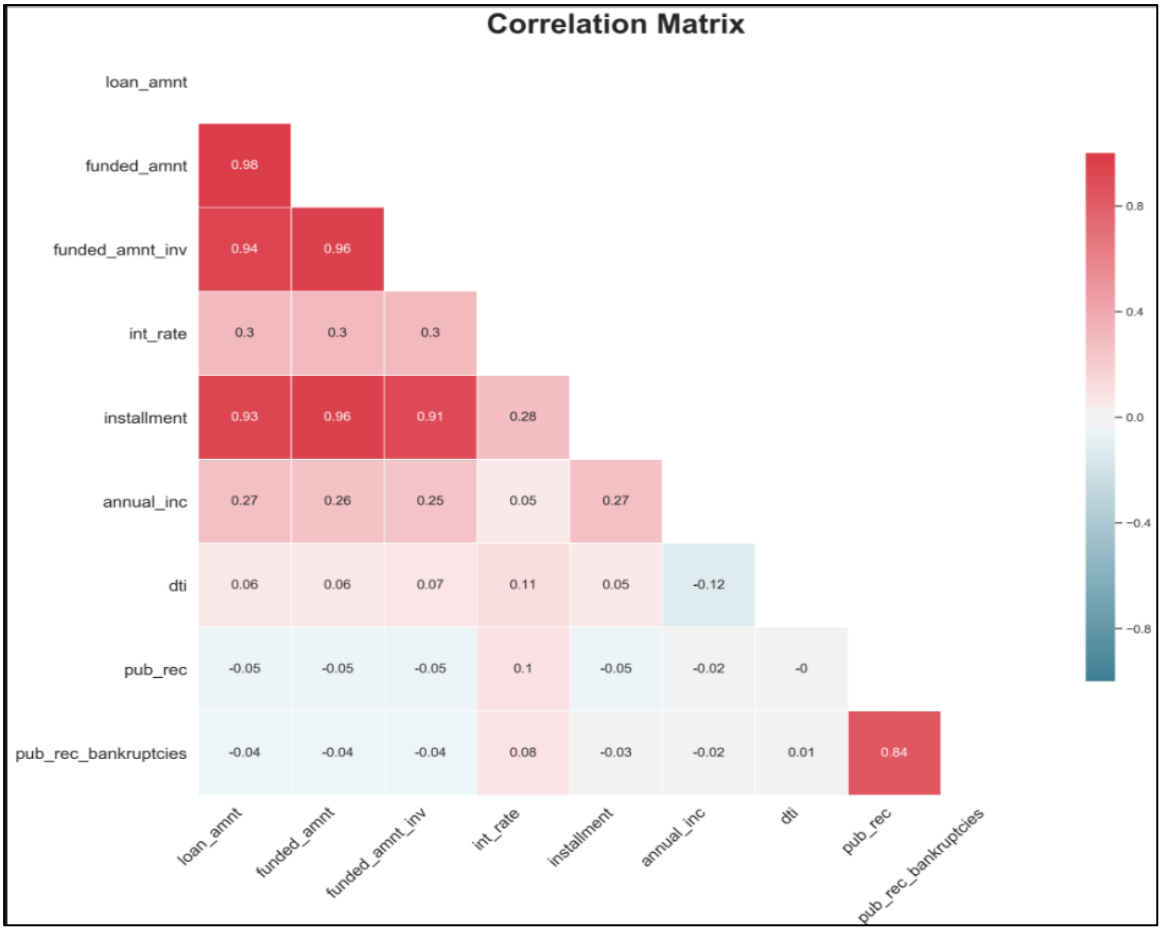
A glimpse of Bivariant Analysis:

```
##Bivariate Analyis - Understanding the correlation of continuous variables
sns.set(style='white')
plt.figure (figsize=(20,20), dpi=300)
df_corr_plot1= df_loan.copy()
#Recall 'col_list_cont: list of columns containing continuous variables'
df_corr_plot1 = df_corr_plot1.loc[:,col_list_cont]
correlation_matrix1 = round(df_corr_plot1.corr(),2)
correlation_matrix1
mask = np.triu(np.ones_like(correlation_matrix1, dtype=np.bool))
f, ax = plt.subplots(figsize=(15, 15), dpi=300)
cmap = sns.diverging_palette(220, 10, as_cmap=True)
ax = sns.heatmap(correlation_matrix1, mask=mask, cmap=cmap, vmin= -1, vmax=1, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5}, annot=True)
plt.xticks(fontsize=15)
plt.yticks(fontsize=15)
plt.title('Correlation Matrix', fontsize=26, fontweight='bold')
plt.setp(ax.get_xticklabels(), rotation=45, horizontalalignment='right')

plt.autoscale()
plt.tight_layout()
plt.show()
```

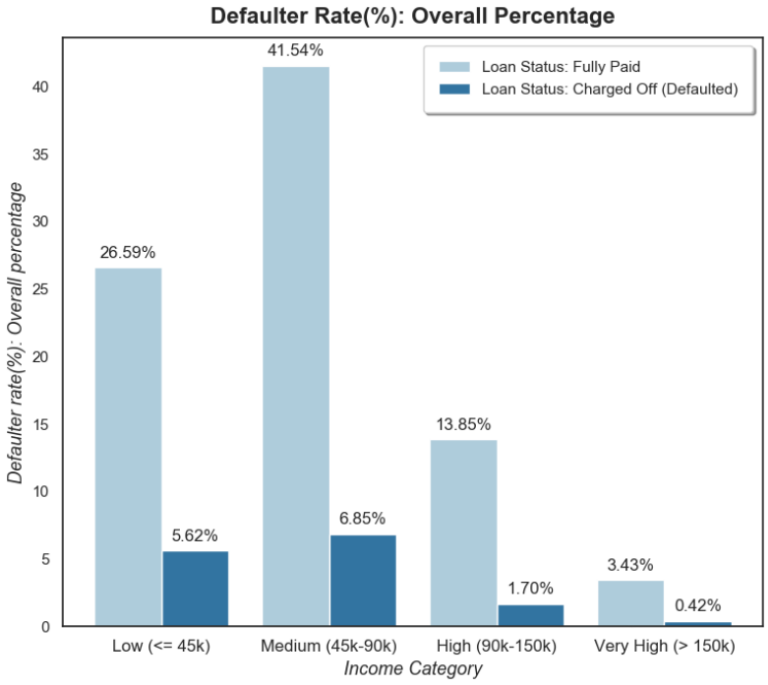
- Correlation matrix has been calculated in '**correlation_matrix1**' using '**df_corr_plot1.corr()'** and rounds the values to 2 decimal places.
- **'mask = np.triu(np.ones_like(correlation_matrix1, dtype=np.bool))**' creates a boolean mask (**mask**) with **True** values in the upper triangle of the correlation matrix.

Decoding the parameters of sns.heatmap:

- ***correlation_matrix1:*** The coefficient values to be visualized.
- ***mask=mask:*** Applies the Boolean mask to hide the upper triangle.
- ***cmap=cmap:*** Uses the defined color palette.
- ***vmin=-1, vmax=1, center=0:*** Sets the color scale from -1 to 1 with the center at 0.
- ***square=True:*** Ensures the heatmap cells are square-shaped.
- ***linewidths=.5:*** Sets the width of the lines between cells to 0.5.
- ***cbar_kws={"shrink": .5}:*** Adjusts the colorbar size.
- ***annot=True:*** Displays the correlation values within the heatmap cells.



- Pearson Correlation plot between continuous variables is retained.
- Numbers are indicating the strength of correlation between variables.



- Loan applicants from 'Low'(<=45K USD) and 'Medium'(45K-90K USD) income group have a greater share of defaulted loan.

Conclusion:

Driving Factors (analysed through plots):

▪ **Loan Term: -** Average Interest rate for defaulted applications is very high with 12.38 % for 36 months and 15.75 % for 60 months term.

▪ **Loan Amount: -** Defaulter rate increases as the requested loan amount increases.

▪ **Annual Income: -** Applicants from 'Low'(<=45K USD) and 'Medium'(45K-90K USD) income group have a greater share of defaulted loans.

▪ **Loan Purpose: -** The top two reasons for loans are debt consolidation and credit card. Such applications should be carefully assessed.